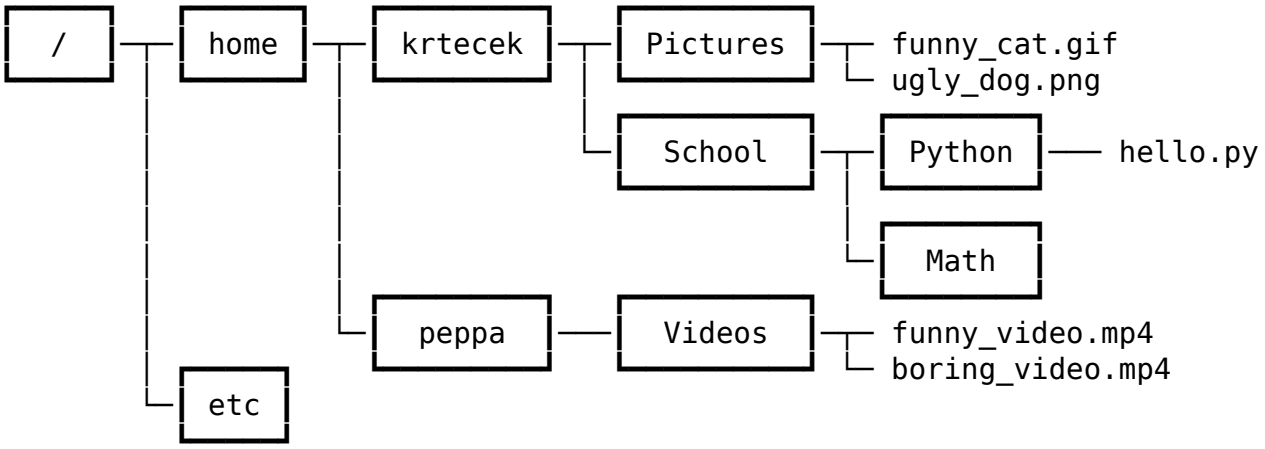
C2184 Úvod do programování v Pythonu

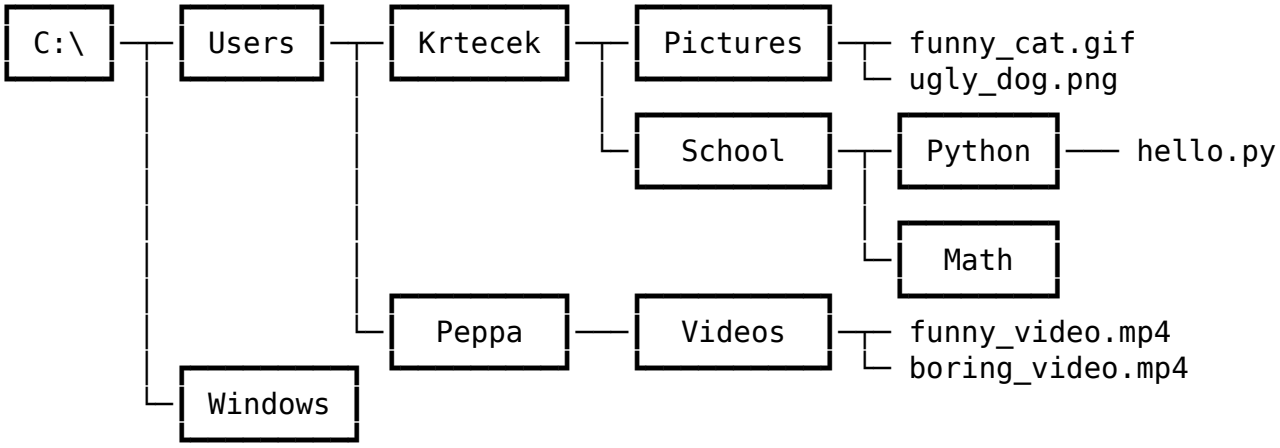
9. Práce se soubory, moduly

Souborový systém

Unix (Linux, MacOS):



Windows:



- **Soubor** (*file*) – sada dat uložených pod konkrétním jménem (např. `funny_cat.gif`)
- **Adresář**, **složka** (*directory, folder*) – speciální soubor odkazující na další

soubory/složky (např. `Pictures`)

- **Kořenový adresář** (*root*) – nejvyšší adresář v systému (v Unixu `/`, ve Windowsu může být víc kořenů: `C:\`, `D:\`…)
- **Domovský adresář** (*home directory*) – nastavený pro každého uživatele (např. `/home/krtecek`, `C:\Users\Krtecek`), často se označuje jako ~
- **Aktuální adresář** (*current working directory*) – kde momentálně jsme, vypisuje se v promptu příkazového řádku
- **Cesta** (*path*) – umístění souboru nebo složky
  - Absolutní cesta – začíná v kořeni souborového systému:

    `/home/krtecek/Pictures/funny_cat.gif` (Unix)

    `C:\Users\Krtecek\Pictures\funny_cat.gif` (Windows)
  - Relativní cesta – bez kořene, začíná v aktuálním adresáři (řekněme `krtecek`):

    `Pictures/funny_cat.gif` (Unix)

    `Pictures\funny_cat.gif` (Windows)
- Speciální znaky v cestě:
  - Oddělovač složek: `/` (Unix) nebo `\` (Windows)
  - `.` – stejná složka
  - `..` – o složku výš (*parent*)

    Příklady:

```
.                            =   /home/krtecek
./Pictures/./funny_cat.gif   =   /home/krtecek/Pictures/funny_cat.gif
..                           =   /home
../peppa/Videos              =   /home/peppa/Videos
../..                        =   /
../../etc                    =   /etc
```

- Příkazový řádek (*bash*, *powershell*, *cmd*) – základní příkazy:
  - `pwd` – vypiš aktuální adresář (*cmd*: `cd`)
  - `ls` – vypiš obsah aktuálního adresáře (*cmd*: `dir`)
  - `cd PATH` – jdi do adresáře PATH (změň aktuální adresář)
- Python – problémy:
  - Rozdíly mezi Unixem a Windowsem (`/` versus `\`…)
  - V řetězcích je `\` speciální znak, takže cestu `C:\Users\Krtecek\Pictures\funny_cat.gif` musíme zapsat:

    `'C:\\Users\\Krtecek\\Pictures\\funny_cat.gif'`

nebo `r'C:\Users\Krtecek\Pictures\funny_cat.gif'` (*raw string*)

Modul pathlib

- Manipulace s cestami
- Smývá rozdíly mezi Unixem a Windowsem
- Základní typ `Path` reprezentuje cestu

[1]:
```
from pathlib import Path
```

- Vytvoření cesty (`Path`):

[2]:
```
Path('/home/krtecek/Pictures/funny_cat.gif')  # Na Windowsu funguje s␣
↪/ i \
```

[2]:
```
PosixPath('/home/krtecek/Pictures/funny_cat.gif')
```

- Domovský a aktuální adresář:

[3]:
```
Path.home()
```

[3]:
```
PosixPath('/home/adam')
```

[4]:
```
Path.cwd()
```

[4]:
```
PosixPath('/home/adam/School/Praca/Python/2022/Python/cviko_09')
```

- Operátor `/` – sřetězení cest (doplní oddělovač `/` nebo `\` podle operačního systému)

[5]:
```
Path.home() / 'Pictures' / 'funny_cat.gif'
```

[5]:
```
PosixPath('/home/adam/Pictures/funny_cat.gif')
```

[6]:
```
Path.home() / Path('Pictures') / Path('funny_cat.gif')
```

[6]:
```
PosixPath('/home/adam/Pictures/funny_cat.gif')
```

- Změna z relativní na absolutní cestu

[7]:
```
Path('../../funny_cat.gif').absolute()
```

[7]:
```
PosixPath('/home/adam/School/Praca/Python/2022/Python/cviko_09/../../
↪funny_cat.g
if')
```

[8]:
```
Path('../../funny_cat.gif').resolve()
```

```
[8]: PosixPath('/home/adam/School/Praca/Python/2022/funny_cat.gif')
```

- Získání částí cesty

```
[9]: Path('/home/krtecek/Pictures/funny_cat.gif').parent
```

```
[9]: PosixPath('/home/krtecek/Pictures')
```

```
[10]: Path('/home/krtecek/Pictures/funny_cat.gif').parent.parent
```

```
[10]: PosixPath('/home/krtecek')
```

```
[11]: Path('/home/krtecek/Pictures/funny_cat.gif').name
```

```
[11]: 'funny_cat.gif'
```

```
[12]: Path('/home/krtecek/Pictures/funny_cat.gif').stem
```

```
[12]: 'funny_cat'
```

```
[13]: Path('/home/krtecek/Pictures/funny_cat.gif').suffix
```

```
[13]: '.gif'
```

```
[14]: Path('/home/krtecek/Pictures/funny_cat.gif').parts
```

```
[14]: ('/', 'home', 'krtecek', 'Pictures', 'funny_cat.gif')
```

Dotazování

```
[15]: p1 = Path('/home/krtecek/Pictures/funny_cat.gif')
      p2 = Path('data/Pictures/funny_cat.gif')
```

```
[16]: p1.is_absolute()
```

```
[16]: True
```

```
[17]: p2.is_absolute()
```

```
[17]: False
```

```
[18]: p2.exists()  # existuje?
```

```
[18]: True
```

```
[19]: p2.is_file()  # existuje a je normální soubor?
```

[19]:
```
True
```

[20]:
```
p2.is_dir()  # existuje a je adresář?
```

[20]:
```
False
```

Výpis adresářů

- Metoda `.iterdir` – iteruje soubory a adresáře v daném adresáři
- Vrací iterátor, který iteruje v neurčitém pořadí – je vhodné použít `sorted()`

[21]:
```
p = Path('.')
p.iterdir()
```

[21]:
```
<generator object Path.iterdir at 0x7feb381995b0>
```

[22]:
```
sorted(p.iterdir())
```

[22]:
```
[PosixPath('09_Soubory.ipynb'),
 PosixPath('09_Soubory.pdf'),
 PosixPath('birthdays.py'),
 PosixPath('collect_sequences.py'),
 PosixPath('count_atoms.py'),
 PosixPath('cviceni_9.ipynb'),
 PosixPath('cviceni_9.pdf'),
 PosixPath('data'),
 PosixPath('du_9.ipynb'),
 PosixPath('du_9.pdf'),
 PosixPath('find.py'),
 PosixPath('header.py'),
 PosixPath('my_modules'),
 PosixPath('paste.py'),
 PosixPath('reseni'),
 PosixPath('sort_districts.py'),
 PosixPath('sum_paper.py'),
 PosixPath('testing.py')]
```

[23]:
```
for file in sorted(p.iterdir()):
    print(file)
```

```
09_Soubory.ipynb
09_Soubory.pdf
birthdays.py
collect_sequences.py
count_atoms.py
cviceni_9.ipynb
```

```
cviceni_9.pdf
data
du_9.ipynb
du_9.pdf
find.py
header.py
my_modules
paste.py
reseni
sort_districts.py
sum_paper.py
testing.py
```

- V některých operačních systémech (Windows) bývá výsledek `.iterdir` už seřazen, ale obecně na to nelze spoléhat –> vždy použijte `sorted()`, pokud chcete seřazené soubory (zejména v domácích úkolech)!

Chytrý výpis adresářů

- Metoda `.glob` – iteruje soubory a adresáře vyhovující zadanému výrazu
  - `*` – libovolný počet znaků (tj. i 0 znaků)
  - `?` – jeden libovolný znak
  - `[A-Za-z]` – jeden znak z výčtu

[24]:
```
# * = všechy soubory
for file in sorted(p.glob('*')):
    print(file)
```

```
09_Soubory.ipynb
09_Soubory.pdf
birthdays.py
collect_sequences.py
count_atoms.py
cviceni_9.ipynb
cviceni_9.pdf
data
du_9.ipynb
du_9.pdf
find.py
header.py
my_modules
paste.py
reseni
sort_districts.py
sum_paper.py
testing.py
```

[25]:
```
# *.pdf = soubory končící .pdf
for file in sorted(p.glob('*.pdf')):
    print(file)
```

```
09_Soubory.pdf
cviceni_9.pdf
du_9.pdf
```

[26]:
```
# *n* = soubory obsahující v názvu n
for file in sorted(p.glob('*n*')):
    print(file)
```

```
09_Soubory.ipynb
collect_sequences.py
count_atoms.py
cviceni_9.ipynb
cviceni_9.pdf
du_9.ipynb
find.py
reseni
testing.py
```

[27]:
```
for file in sorted(p.glob('data/Pictures/?ad_giraffe.*')):
    print(file)
```

```
data/Pictures/bad_giraffe.jpg
data/Pictures/mad_giraffe.png
data/Pictures/sad_giraffe.jpg
```

[28]:
```
for file in sorted(p.glob('data/Pictures/[bm]ad_giraffe.*')):
    print(file)
```

```
data/Pictures/bad_giraffe.jpg
data/Pictures/mad_giraffe.png
```

- Metoda `.rglob` – rekurzivní glob, tj. prohledává i podadresáře, podpodadresáře atd.

[29]:
```
for file in sorted(p.rglob('*.png')):
    print(file)
```

```
data/Pictures/mad_giraffe.png
```

Manipulace se souborovým systémem

[31]:
```
directory = Path('data/new_dir')
```

- Vytvoření adresáře

[32]:
```
directory.mkdir(exist_ok=True)  # make directory
```

- Přesun souboru

[33]:
```
Path('data/Pictures/funny_cat.gif').replace(directory / 'funny_cat2.
↪gif')
```

[33]:
```
PosixPath('data/new_dir/funny_cat2.gif')
```

[34]:
```
(directory / 'funny_cat2.gif').replace('data/Pictures/funny_cat.gif')
```

[34]:
```
PosixPath('data/Pictures/funny_cat.gif')
```

- Smazání souboru

[35]:
```
Path('ugly_dog.jpg').unlink(missing_ok=True)
```

- Smazání prázdného adresáře

[36]:
```
Path('data/new_dir').rmdir()  # remove directory
```

Moduly `os` a `os.path`

- Funkce závislé na operačním systému
- Většina funkcí jsou pouze zastaralé verze funkcí z modulu `pathlib`
- Funkce `os.chdir()` – změň aktuální adresář

[37]:
```
import os

print(Path.cwd())
os.chdir('data')
print(Path.cwd())
os.chdir('..')
print(Path.cwd())
```

```
/home/adam/School/Praca/Python/2022/Python/cviko_09
/home/adam/School/Praca/Python/2022/Python/cviko_09/data
/home/adam/School/Praca/Python/2022/Python/cviko_09
```

Modul `shutil`

- Více možností než `pathlib` a `os`

[38]:
```
import shutil

Path('stuff').mkdir(exist_ok=True)                    # Vytvoř␣
↪adresář stuff
```

```
Path('stuff', 'foo').mkdir(exist_ok=True)                # Vytvoř␣
  ↪adresář stuff/foo
Path('stuff', 'file1.txt').write_text('blabla')          # Vytvoř␣
  ↪soubor stuff/file1.txt
Path('stuff', 'foo', 'file2.txt').write_text('blabla')  # Vytvoř␣
  ↪soubor stuff/foo/file2.txt
```

[38]:
```
6
```

[39]:
```
shutil.rmtree('stuff')  # Smaž adresář stuff a všechno v něm
```

[40]:
```
shutil.copy('data/Pictures/funny_cat.gif', Path.home()/'CLICK_HERE.
  ↪gif')  # Zkopíruj soubor
```

[40]:
```
PosixPath('/home/adam/CLICK_HERE.gif')
```

[41]:
```
shutil.copytree('data/Pictures', Path.home()/'magic_inside')  #␣
  ↪Zkopíruj adresář a všechno v něm
```

[41]:
```
PosixPath('/home/adam/magic_inside')
```

[42]:
```
shutil.make_archive('stuff_archive', 'zip', 'stuff')  # Komprimuj␣
  ↪adresář stuff do stuff_archive.zip
```

[42]:
```
'/home/adam/School/Praca/Python/2022/Python/cviko_09/stuff_archive.
  ↪zip'
```

Čtení a zápis do souborů

Základní rozdělení formátů

- **Textové** (TXT, HTML, CSV, JSON…) – soubor tvořen posloupností znaků

  Bohuš je rázovitý vesnický chlapík, který žije sám v rozpadající se chalupě. Pracuje jako dřevař, většinu času však se svými kumpány tráví popíjením slivovice a zahálčivým povalováním v trávě. Jeho poklidný a vcelku spokojený život se změní v okamžiku, kdy se objeví elegantní advokát Ulrich, který překvapenému Bohušovi oznámí, že zdědil obrovský majetek.

- **Binární** (ZIP, JPG, DOCX…) – soubor tvořen posloupností bajtů

  `x␣␣E␣␣␣␣L␣/lX␣ə10r␣m-[ö3␣5␣␣␣H␣␣␣j+□/␣/␣a␣6␣␣R␣␣␣␣/␣;␣\'␣-n␣ ;␣␣u␣␣␣␣M␣2B␣␣␣␣#"␣f␣␣␣7␣␣␣"-␣,␣␣␣=␣␣[␣␣2apk␣p␣␣␣␣/␣)V!<␣␣␣␣6 ␣␣␣h|Qb1L␣␣*␣␣␣7␣x␣␣E␣␣␣␣L␣/lX␣ə10r␣m-[ö3␣`

- Poznámka: Textový soubor je vlastně binární soubor, kde každý bajt nebo skupina bajtů kóduje nějaký znak. Rozdíl je pouze v tom, jak budeme soubor číst.

Základní operace se souborem

- Otevření (*open*)
- Čtení (*read*)
- Zápis (*write*)
- Zavření (*close*) = uložení změn na disk

Otevření a zavření souboru v Pythonu

- Explicitně – hrozí, že zapomeneme soubor zavřít a změny se neuloží – nedoporučuje se

```
f = open('my_file.txt')
...  # Pracujeme se souborem
...  # Pracujeme se souborem
f.close()
```

- Pomocí bloku `with` – doporučený způsob (soubor se zavře automaticky na konci bloku, i kdyby nastala výjimka)

```
with open('my_file.txt') as f:
    ...  # Pracujeme se souborem
    ...  # Pracujeme se souborem
    # Na konci bloku with se soubor automaticky zavře
```

[43]:
```
with open('data/CSFD/Dedictvi.txt', mode='r', encoding='utf8') as f:
    print(f)
```

```
<_io.TextIOWrapper name='data/CSFD/Dedictvi.txt' mode='r'␣
↪encoding='utf8'>
```

Funkce `open`

- Argumenty:
  - Cesta k souboru (typ `str` nebo `Path`)
  - Mód, kódovaní...
- Návratová hodnota:
  - Souborový objekt (*file object*, *file handle*) – popisuje, který soubor je otevřený, mód, kódování, ukazatel na konkrétní pozici v souboru...

Mód otevření textového souboru (*mode*)

- `'r'`: pro čtení (*read*) – defaultní mód
  - soubor existuje -> čte od začátku
  - soubor neexistuje -> `FileNotFoundError`

- `'w'`: pro zápis (*write*)
  - soubor existuje –> přemaže a zapisuje od začátku!
  - soubor neexistuje –> vytvoří ho
- `'a'`: pro zápis na konec (*append*)
  - soubor existuje –> doplňuje na konec
  - soubor neexistuje –> vytvoří ho a zapisuje od začátku

Další módy:

- `'x'`: pro zápis
  - soubor existuje –> `FileExistsError`
  - soubor neexistuje –> vytvoří ho
- `'r+'` pro čtení a zápis
  - soubor existuje –> čte/zapisuje od začátku
  - soubor neexistuje –> `FileNotFoundException`
- `'w+'` pro čtení a zápis
  - soubor existuje –> přemaže a čte/zapisuje od začátku
  - soubor neexistuje –> vytvoří ho

Kódování souborů (*encoding*)

- V souborech reálně nejsou uloženy znaky, ale pouze bajty.
- Kódování popisuje, jak se znaky zakódují do bajtů (*encode*) a zpátky (*decode*).
- Příklady kódování:
  - **`UTF-8 (utf8)`** – nejmodernější, dokáže zakódovat celou znakovou sadu *Unicode* (některé znaky se ukládají na více bajtů)
  - `windows-1250` (`cp1250`) – středoevropské
  - `windows-1252` (`cp1252`) – západoevropské
- Znaky ze sady *ASCII* (prvních 128 znaků z Unicode) se kódují stejně ve většině kódování
- Defaultní kódování závisí na systému:

[44]:
```
import locale

locale.getpreferredencoding()
```

[44]:
```
'UTF-8'
```

- Příklad špatného rozkódování:

  - Soubor v `windows-1250` rozkódovaný jako `windows-1252`:

```
Povìst vypráví o legendárním bojovníkovi,
jeho umìní Kung-Fu bylo mýtické...
Cestoval napøíè zemí a hledal nepøátele.

Koukám, že rád žvýkáš.
Možná bys mìl vyzkoušet mojí pìst!
```

  - Správné rozkódování:

```
Pověst vypráví o legendárním bojovníkovi,
jehož umění Kung-Fu bylo mýtické...
Cestoval napříč zemí a hledal nepřátele.

Koukám, že rád žvýkáš.
Možná bys měl vyzkoušet mojí pěst!
```

- VSCode umožňuje načítat/ukládat soubory v různých kódováních (na dolní liště vpravo).

Čtení z textového souboru

- Funguje pouze v módech **r**, r+, w+
- Metoda `.read` přečte celý soubor (bez argumentu) nebo daný počet znaků (s argumentem)
  - Vrátí načtený řetězec
- Po zavření souboru už nelze číst

[45]:
```
with open('data/CSFD/Dedictvi.txt', mode='r', encoding = 'utf8') as f:
    text = f.read()
text
```

[45]:
```
'Bohuš je rázovitý vesnický chlapík, který žije sám v rozpadající se
chalupě.\nPracuje jako dřevař, většinu času však se svými kumpány␣
  ↪trávípopíjením\nslivovice a zahálčivým povalováním v trávě. Jeho poklidný␣
  ↪a vcelkuspokojený\nživot se změní v okamžiku, kdy se objeví elegantní advokát␣
  ↪Ulrich,který\npřekvapenému Bohušovi oznámí, že zdědil obrovský majetek.'
```

[46]:
```
with open('data/CSFD/Dedictvi.txt', mode='r', encoding = 'utf8') as f:
    first20 = f.read(20)
    next10 = f.read(10)
    rest = f.read()
print(first20)
print('-----')
```

```
print(next10)
print('-----')
print(rest)
```

```
Bohuš je rázovitý ve
-----
snický chl
-----
apík, který žije sám v rozpadající se chalupě.
Pracuje jako dřevař, většinu času však se svými kumpány tráví popíjením
slivovice a zahálčivým povalováním v trávě. Jeho poklidný a vcelku␣
  ↪spokojený
život se změní v okamžiku, kdy se objeví elegantní advokát Ulrich,␣
  ↪který
překvapenému Bohušovi oznámí, že zdědil obrovský majetek.
```

- Metoda `.readline` načte jeden řádek ze souboru jako řetězec.
- Pokud jsme už na konci souboru, vrátí se prázdný řetězec `''`.

[47]:
```
with open('data/a.txt', encoding = 'utf8') as f:
    line1 = f.readline()
    line2 = f.readline()
    line3 = f.readline()
    line4 = f.readline()  # už jsme na konci souboru
    line5 = f.readline()  # už jsme na konci souboru
```

[48]:
```
[line1, line2, line3, line4, line5]
```

[48]:
```
['Alice\n', 'Bob\n', 'Cyril', '', '']
```

- Řádek se načítá vždy se znakem nového řádku na konci.
- Tohoto se zbavíme pomocí metody `.rstrip`.

[49]:
```
line1
```

[49]:
```
'Alice\n'
```

[50]:
```
line1.rstrip()
```

[50]:
```
'Alice'
```

- Metoda `.readlines` načte všechny řádky (od aktuální pozice) jako seznam

[51]:
```
with open('data/a.txt', encoding = 'utf8') as f:
    lines = f.readlines()
lines
```

[51]:
```
['Alice\n', 'Bob\n', 'Cyril']
```

Čtení souboru pomocí `for`

- Souborový objekt je zároveň iterátorem – iteruje soubor po řádcích

[52]:
```
with open('data/a.txt', mode='r', encoding = 'utf8') as f:
    for line in f:
        print('>', line.rstrip())
```

```
> Alice
> Bob
> Cyril
```

- Objekt typu `Path` umí načíst celý soubor (jako `.read`, ale otevře a zavře ho za nás)

[53]:
```
Path('data/a.txt').read_text(encoding='utf8')
```

[53]:
```
'Alice\nBob\nCyril'
```

Zápis do textového souboru

- Funguje pouze v módech **`w`**, **`a`**, `x`, `r+`, `w+`
- Metoda `.write` zapíše řetězec do souboru
  - Argument musí být vždy pouze řetězec
  - Vrátí počet zapsaných znaků
- Na konec se nevkládá automaticky nový řádek, jako u `print`
- Po zavření souboru už nelze zapisovat

[54]:
```
with open('data/novy.txt', mode='w', encoding='utf8') as f:
    f.write('Alice')
    f.write('Bob')
    f.write('Cyril')
```

[55]:
```
print(Path('data/novy.txt').read_text(encoding='utf8'))
```

```
AliceBobCyril
```

- Funkci `print` lze přesměrovat do souboru pomocí parametru `file`:

[56]:
```
with open('data/novy.txt', mode='w', encoding='utf8') as f:
    print('Alice', 10, True, file=f)
    print('Bob', 20, False, file=f)
```

[57]:
```
print(Path('data/novy.txt').read_text(encoding='utf8'))
```

```
Alice 10 True
Bob 20 False
```

- Objekt typu `Path` umí zapsat celý soubor (jako `.write`, ale otevře a zavře ho za nás)

[58]:
```
Path('data/novy.txt').write_text('blablabla', encoding='utf8')
```

[58]: 9

[59]:
```
print(Path('data/novy.txt').read_text(encoding='utf8'))
```

```
blablabla
```

- Můžeme otevřít několik souborů současně

[60]:
```
with open('data/a.txt') as fa:
    with open('data/b.txt') as fb:
        with open('data/ab.txt', mode='w') as fab:
            for line_a, line_b in zip(fa, fb):
                print(line_a.rstrip(), line_b.rstrip(), file=fab)
```

[61]:
```
print(Path('data/a.txt').read_text())
print('------')
print(Path('data/b.txt').read_text())
print('------')
print(Path('data/ab.txt').read_text())
```

```
Alice
Bob
Cyril
------
10
20
30
------
Alice 10
Bob 20
Cyril 30
```

Metody `tell` a `seek`

- Metoda `.tell` vrací aktuální pozici ukazatele, `.seek` nastavuje pozici ukazatele

(Není nutné umět.)

[62]:
```
with open('data/a.txt') as f:
    print(f.tell())
    print(f.readline().rstrip())
    print(f.tell())
    print(f.readline().rstrip())
    print(f.tell())
    print(f.readline().rstrip())
    print(f.tell())
```

```
0
Alice
6
Bob
10
Cyril
15
```

[63]:
```
with open('data/a.txt') as f:
    f.seek(6)
    print(f.readline().rstrip())
    f.seek(0)
    print(f.readline().rstrip())
```

```
Bob
Alice
```

Speciální souborové objekty

- `sys.stdin` – čte ze standardního vstupu
- `sys.stout` – zapisuje na standardní výstup
- `sys.stderr` – zapisuje na standardní chybový výstup

[64]:
```
import sys

sys.stdout.write('ahoj')
```

```
ahoj
```

[64]: 4

Ukončování řádků

- Unix (Linux a MacOS) ukončuje řádky jedním znakem `'\n'` (LF = *line-feed*)
- Windows ukončuje řádky dvojící znaků `'\r\n'` (CRLF = *carriage-return line-feed*)
- Python tento rozdíl skrývá

  - `'\n'` i `'\r\n'` se načte jako `'\n'`
  - `'\n'` se zapíše jako `'\n'` v Unixu, jako `'\r\n'` ve Windowsu
- VSCode vpravo dole vypisuje kódování a ukončování řádku, umožňuje snadnou konverzi

Zkoušíme otevřít soubor

- *Ask for forgiveness, not for permission.*
- Nezjišťujeme, jestli soubor existuje a lze ho otevřít. Prostě ho zkusíme otevřít.

[65]:
```
try:
    with open('neexistujici_soubor.txt') as f:
        print(f.read())
except FileNotFoundError:
    print('Soubor neexistuje.')
except PermissionError:
    print('Nemáš právo číst soubor.')
except OSError:
    print('Soubor se nepodařilo otevřít.')
```

```
Soubor neexistuje.
```

- `OSError` je obecnější typ výjimky, zahrnuje `FileNotFoundError`, `FileExistsError`, `PermissionError`...

Na co si dát pozor

- Kódování souborů
- Řádky vždy končí bílými znaky (`\n`), proto je vhodné použití metod `.rstrip` nebo `.strip`
- Když soubor zavřeme, už s ním nemůžeme dále pracovat (read, write); jedině že si ho znovu otevřeme
- Pokud soubor otevíráme v módu `'w'`, hrozí ztráta dat – současný soubor je ihned nenávratně přepsán novým prázdným souborem

Binární soubory

- Místo řetězců (typ `str`) čteme a zapisujeme bajty (typ `bytes`)
- Binární módy otevření: **`'rb'`**, **`'wb'`**, **`'ab'`**, **`'xb'`**, **`'r+b'`**, **`'w+b'`**
- Metoda `.read` vrací typ `bytes`
- Metoda `.write` bere typ `bytes`

[66]:
```
with open('data/Pictures/funny_cat.gif', mode='r+b') as f:
    content = f.read(100)
```

```
content
```

[66]:
```
b'GIF89a\xc8\x00\xfa\x00\xf7\xfd\x00\x16\x0b\x04\x1a\x0e\x0c\x1a\x14\x14\x1b\x12
\x0c\x1d\x18\x1a"\x1c\x1d#\x18\x14$\x15\x0b%\x1e!&␣
↪#*AI+\x1d\x1a+#\x1c+%$,!\x15-
\x1a\x0c-\x1c\x14-&).)).*%.JT01B2,*2NZ3%\x1d3+%4"\x134)\x1b61-'
```

- Konverze mezi `str` a `bytes` pomocí metod `.encode` a `.decode`:

[67]:
```
'Náhodná věta'.encode()  # UTF-8 kódování
```

[67]:
```
b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'
```

[68]:
```
'Náhodná věta'.encode('cp1250')  # Windows středoevropské kódování
```

[68]:
```
b'N\xe1hodn\xe1 v\xecta'
```

[69]:
```
b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'.decode()
```

[69]:
```
'Náhodná věta'
```

[70]:
```
b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'.decode('cp1250')
```

[70]:
```
'NÃˇhodnÃˇ vÄ›ta'
```

Moduly

Modul (*module*)

- Soubor funkcí a proměnných, které lze importovat

[71]:
```
import math
```

[72]:
```
math
```

[72]:
```
<module 'math' (built-in)>
```

[73]:
```
math.cos(0)
```

[73]:
```
1.0
```

[74]:
```
math.pi
```

[74]:
```
3.141592653589793
```

Balíček (*package*)

- Modul obsahující další moduly (složka s moduly)

[75]:
```
import os
```

[76]:
```
os
```

[76]:
```
<module 'os' from '/usr/lib/python3.10/os.py'>
```

[77]:
```
os.path
```

[77]:
```
<module 'posixpath' from '/usr/lib/python3.10/posixpath.py'>
```

Import pomocí `from` a `as`

- `from` – importujeme modul z balíčku nebo proměnnou/funkci z modulu
- `as` – přejmenování importovaného modulu/proměnné/funkce

[78]:
```
from os import path

path.exists('data')
```

[78]:
```
True
```

[79]:
```
from math import factorial, pi

factorial(6) + pi
```

[79]:
```
723.1415926535898
```

[80]:
```
import numpy as np

np.linspace(0, 1, 11)
```

[80]:
```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])
```

[81]:
```
from math import factorial as fact, pi as PI

fact(6) + PI
```

[81]:
```
723.1415926535898
```

Spuštění modulu jako skriptu

- Z příkazového řádku pomocí přepínače `-m`:

```
$ python -m doctest
```

Zdroje modulů / balíčků

- Standardní knihovna Pythonu
  - Moduly přítomné při instalaci
  - `math`, `pathlib`, `sys`...
  - https://docs.python.org/3.10/library/
- PyPI – *Python Package Index*
  - Moduly doinstalovatelné např. pomocí nástoje `pip`
  - `numpy`, `sklearn`...
  - https://pypi.org/
- Vlastní moduly
  - Každý pythonovský skript lze importovat jako modul

Pip

- Modul sloužící ke stáhnutí a doinstalování balíčků z PyPI
- Spouštíme ho vždy z příkazového řádku pomocí `-m` (nesmí se importovat)

```
$ python -m pip install numpy  # Nainstaluj balíček numpy

$ python -m pip show numpy     # Vypiš verzi a info o nainstalovaném balíčku

$ python -m pip list           # Vypiš seznam nainstalovaných balíčků

$ python -m pip help           # Vypiš nápovědu k pipu
```

- V učebně C04/118 `pip` nefunguje (balíčky instaluje admin), ale lze instalovat balíčky v rámci virtuálního prostředí:

```
# Vytvoření virt. prostředí:
$ module add anaconda3
$ python -m venv ~/venv

# Další použití virt. prostředí:
$ . ~/venv/bin/activate
$ python -m pip install some_package
$ python ...
```

Vlastní moduly

- 1 soubor `.py` = 1 modul
- Máme dlouhý program –> můžeme ho rozdělit na více modulů

- Z hlavního skriptu (`__main__`) pak importujeme ostatní moduly
- Stejný modul lze importovat do více skriptů nebo do dalších modulů

Soubor `my_modules/areas.py`:

```
"""
Module for computation of areas of geometric shapes.
"""

import math

def rectangle_area(a: float, b:float) -> float:
    """Return the area of a rectangle with sides `a`, `b`."""
    return a * b

def square_area(a: float) -> float:
    """Return the area of a square with side `a`."""
    return rectangle_area(a, a)

def circle_area(r: float) -> float:
    """Return the area of a circle with radius `r`."""
    return math.pi * r**2
```

[82]:
```
from my_modules import areas

areas
```

[82]:
```
<module 'my_modules.areas' from
'/home/adam/School/Praca/Python/2022/Python/cviko_09/my_modules/areas.
  ↪py'>
```

[83]:
```
areas.rectangle_area(7, 6)
```

[83]:
```
42
```

`if __name__ == '__main__':`

- Tento blok se vykoná, pouze když modul spouštíme jako skript.
- Když modul importujeme, tento blok se nevykoná.
- (Toto funguje díky tomu, že pokud je modul importován, v magické proměnné `__name__` je název modulu; pokud je spouštěn jako skript, v proměnné `__name__` je `'__main__'`.)

Soubor `my_modules/fibonacci.py`:

```
"""
Module for generating the Fibonacci numbers.
"""
```

```
def fib(n: int) -> list[int]:
    """Return the first `n` Fibonacci numbers."""
    result = []
    a, b = 1, 1
    while len(result) < n:
        result.append(a)
        a, b = b, a+b
    return result

def main() -> None:
    """Interactive printing of the Fibonacci numbers."""
    n_numbers = int(input('Required number of printed elements: '))
    numbers = fib(n_numbers)
    print(*numbers, sep=', ')

if __name__ == '__main__':
    main()
```

- Import:

[84]:
```
from my_modules import fibonacci as fib
```

[85]:
```
fib.fib(10)
```

[85]:
```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

- Spuštění z příkazového řádku:

  `$ python my_modules/fibonacci.py`

  ```
  Zadej požadovaný počet prvků: 10
  1, 1, 2, 3, 5, 8, 13, 21, 34, 55
  ```

  nebo

  `$ python -m my_modules.fibonacci`

  ```
  Zadej požadovaný počet prvků: 8
  1, 1, 2, 3, 5, 8, 13, 21
  ```

Typická struktura pythonovského programu

- Docstring – co tento modul/skript dělá
- Importy
- Konstanty
- Funkce
- `if __name__ == '__main__':  main()`

Přehled užitečných modulů

- Standardní knihovna: https://docs.python.org/3.10/library/
- Python Package Index: https://pypi.org/

Modul `math`

- Matematické funkce

[86]:
```
import math

math.comb(15, 3)
```

[86]:
```
455
```

Modul `cmath`

- Matematické funkce nad komplexními čísly

[87]:
```
import cmath

cmath.sqrt(-1)
```

[87]:
```
1j
```

[88]:
```
cmath.sin(2+1j)
```

[88]:
```
(1.4031192506220405-0.4890562590412937j)
```

Modul `statistics`

- Statistické funkce

[89]:
```
import statistics

statistics.median([8, 2, 5, 1, 3])
```

[89]:
```
3
```

[90]:
```
statistics.correlation([1, 2, 3, 4], [5, 6, 10, 12])
```

[90]:
```
0.9768308314557045
```

Modul `random`

- Generování pseudo-náhodných čísel

[91]:
```
import random
```

[92]:
```
random.random()  # Náhodné reálné číslo z intervalu [0, 1)
```

[92]:
```
0.12980814405144891
```

[93]:
```
random.randint(0, 100)  # Náhodné celé číslo z intervalu [0, 100]
```

[93]:
```
41
```

[94]:
```
random.choice(['červená', 'zelená', 'modrá', 'černá'])  # Náhodný␣
↪výběr
```

[94]:
```
'zelená'
```

Modul `datetime`

- Práce s časovými údaji
- Základní typy:
  - `datetime` – datum a čas
  - `timedelta` – trvání
  - `date` – datum
  - `time` – čas
  - `timezone` – časová zóna

[95]:
```
from datetime import datetime, timedelta

start = datetime.now()
math.factorial(500_000)
end = datetime.now()
calculation_time = end - start
print(calculation_time)
```

```
0:00:02.249318
```

[96]:
```
today = datetime.now().date()
in_a_week = today + timedelta(weeks=1)
print(f'Today is: {today}')
print(f'In a week it will be: {in_a_week}')
```

```
Today is: 2022-11-14
In a week it will be: 2022-11-21
```

- Formátování času

[97]:
```
datetime.now().strftime('%A, %d %B %Y, %H:%M:%S')
```

[97]:
```
'Monday, 14 November 2022, 12:18:05'
```

- Parsování času

[98]:
```
datetime.strptime('1. 7. 2000, 13:30:05', '%d. %m. %Y, %H:%M:%S')
```

[98]:
```
datetime.datetime(2000, 7, 1, 13, 30, 5)
```

Modul `itertools`

- Různé možnosti iterování

[99]:
```
import itertools
```

[100]:
```
for pair in itertools.combinations(['bílá', 'zelená', 'modrá'], 2):
    print(*pair)
```

```
bílá zelená
bílá modrá
zelená modrá
```

[101]:
```
for pair in itertools.permutations(['bílá', 'zelená', 'modrá'], 2):
    print(*pair)
```

```
bílá zelená
bílá modrá
zelená bílá
zelená modrá
modrá bílá
modrá zelená
```

[102]:
```
for letter, number in itertools.zip_longest('ABCDE', [1, 2, 3],␣
↪fillvalue='?'):
    print(letter, number)
```

```
A 1
B 2
C 3
D ?
E ?
```

Moduly `pathlib`, `shutil`, `os.path`

- Funkce na práci se souborovým systémem (viz úvod přednášky)

Modul os

- Funkce závislé na operačním systému

[103]:
```
import os

os.cpu_count()
```

[103]: 8

[104]:
```
os.getenv('USER')
```

[104]: 'adam'

[105]:
```
os.chdir('data')
os.chdir('..')
```

Modul sys

- Funkce závislé na systému

[106]:
```
import sys

sys.version  # Verze interpretru
```

[106]: '3.10.6 (main, Nov 2 2022, 18:53:38) [GCC 11.3.0]'

- `sys.argv` – seznam argumentů z příkazového řádku
  - Nultý argument = název skriptu
- Soubor `my_modules/suma.py`:

```
import sys

print('sys.argv:', sys.argv)
numbers = [int(x) for x in sys.argv[1:]]
the_sum = sum(numbers)
print(the_sum)
```

- Spustíme z příkazového řádku:

```
$ python my_modules/sum.py 1 5 8
```

```
['sum.py', '1', '5', '8']
14
```

- V praxi je omnoho praktičtější použít modul `argparse` (v příští přednášce)

Další užitečné moduly

(Některé si ještě ukážeme ve zbytku semestru)

- argparse
- json
- csv
- pickle
- requests
- venv
- mypy
- doctest
- subprocess
- email
- re
- codecs
- warnings
- numpy
- matplotlib
- PIL
- sklearn
- pandas
- …